

IN THE CLAIMS:

1. – 14. (CANCELLED)

15. (PREVIOUSLY PRESENTED) A process for generating a micro-processor instruction set extension for a processor application, comprising:

generating a data flow graph $G(V,E)$ of nodes V representing primitive operations of the processor application and edges E representing data dependencies of said application;

evaluating subgraphs S of $G(V,E)$ as candidates for an instruction set extension, each said subgraph S having a number of inputs $IN(S)$ and a number of outputs $OUT(S)$, said instruction set extension having a number of available register-file read ports N_{in} and a number of available register-file write ports N_{out} ;

wherein said evaluating a subgraph S includes,

if $OUT(S)$ is less than or equal to N_{out} , and

if S is convex, and

if $IN(S)$ is less than or equal to N_{in} ,

then identifying S as a candidate for transformation into an instruction set extension, else disregarding S as a candidate for transformation into an instruction set extension;

wherein S is convex when no path exists from a node in S to another node in S when said path involves a node that is not in S ;

evaluating said identified candidates using a function $M(S)$ as a measure of merit;

transforming said instruction set by adding an instruction set extension representing said identified candidate to said instruction set if said candidate satisfies said function $M(S)$.

16. (PREVIOUSLY PRESENTED) The process of claim 15, further comprising:

performing said evaluating of subgraphs S for a plurality all subgraphs S in a plurality of data flow graphs G representing all basic blocks of said processor application;

selecting a number j of identified candidates satisfying said evaluation to form a candidate set S_j which maximizes $\sum_j M(S_j)$; and

transforming said candidate set S_j into instruction set extensions for said processor

application.

17. (PREVIOUSLY PRESENTED) The process of claim 15, comprising:
performing said evaluation of subgraphs S for a plurality of subgraphs S in a single basic block of said processor operation;
identifying a single optimal S in said plurality according to said function M(S); and
transforming said instruction set by adding an instruction set extension representing only said optimal S.

18. (PREVIOUSLY PRESENTED) The process of claim 17, comprising:
identifying an optimal set of non-overlapping subgraphs S in a plurality of basic blocks of said processor operation according to said function M(S); and
transforming said instruction set by adding one or more instruction set extensions representing said optimal set of non-overlapping subgraphs.

19. (CURRENTLY AMENDED) The process of claim 17, comprising:
identifying a ~~near-optimal~~ set of non-overlapping subgraphs S in a plurality of basic blocks of said processor operation according to said function M(S); and
transforming said instruction set by adding one or more instruction set extensions representing said ~~near-optimal~~ set of non-overlapping subgraphs.

20. (PREVIOUSLY PRESENTED) The process of claim 17, comprising:
performing a topological sort on G;
ordering nodes of G such that if G contains an edge (u,v) then u appears after v in said ordering; and
utilizing a recursive search function based on said ordering.

21. (PREVIOUSLY PRESENTED) A system for generating a micro-processor instruction set extension for a processor application, comprising:
computer means for generating a data flow graph G(V,E) of nodes V representing

primitive operations of the processor application and edges E representing data dependencies of said application;

computer means for evaluating subgraphs S of $G(V,E)$ as candidates for an instruction set extension, each said subgraph S having a number of inputs $IN(S)$ and a number of outputs $OUT(S)$, said instruction set extension having a number of available register-file read ports N_{in} and a number of available register-file write ports N_{out} ;

wherein said evaluating a subgraph S includes,

if $OUT(S)$ is less than or equal to N_{out} , and

if S is convex, and

if $IN(S)$ is less than or equal to N_{in} ,

then identifying S as a candidate for transformation into an instruction set extension, else disregarding S as a candidate for transformation into an instruction set extension; wherein S is convex when no path exists from a node in S to another node in S when said path involves a node that is not in S ;

computer means for evaluating said identified candidates using a function $M(S)$ as a measure of merit; and

computer means for transforming said instruction set by adding an instruction set extension representing said identified candidate to said instruction set if said candidate satisfies said function $M(S)$.